

Tools and Techniques for Analyzing Type Enforcement Policies

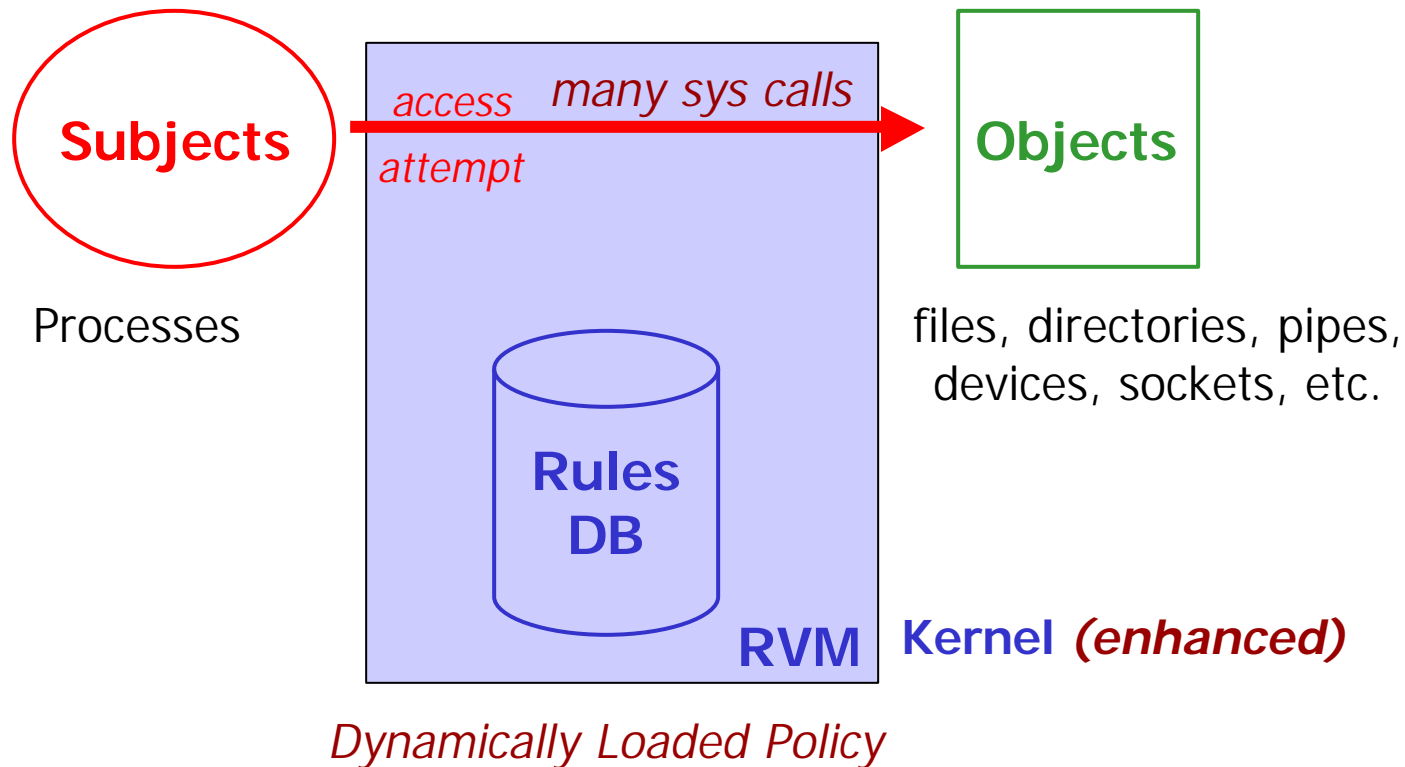


Improving Security Enhanced Linux

Frank Mayer
Tresys Technology
mayerf@tresys.com

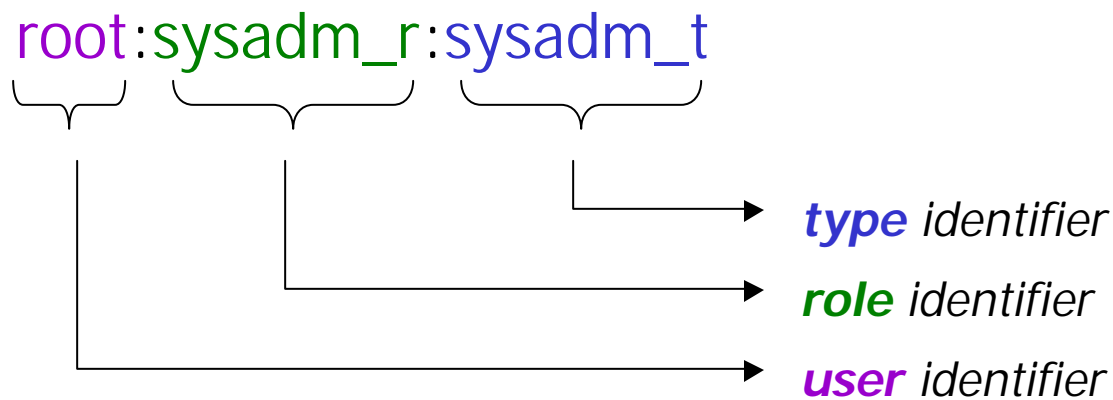
Annual Computer Security Conference
Las Vegas, NV
December, 2003

SE Linux Access Control



SELinux Access Control Attributes

- SE Linux assigns subject and objects a security context:



- Security context is only access control attribute in SE Linux
- Type is principal access control attribute

SE Linux Provides Great Granularity

- Types: effectively unlimited
- Object Classes: user-mode extendable
 - SE Linux currently defines 29 kernel object classes

1) security	2) process	3) system	4) capability
5) filesystem	6) file	7) dir	8) fd
9) lnk_file	10) chr_file	11) blk_file	12) sock_file
13) fifo_file	14) socket	15) tcp_socket	16) udp_socket
17) msgq	18) sem	19) msg	20) shm
21) ipc	22) node	23) netif	24) netlink_socket
25) packet_socket	26) key_socket	27) rawip_socket	
28) unix_stream_socket	29) unix_dgram_socket		
 - Each with their own fine-grained permissions, e.g., file class:

ioctl	read	write	create
getattr	setattr	lock	relabelfrom
relabelto	append	unlink	link
rename	execute	swapon	quotaon
mounton	execute_no_trans	entrypoint	

Example TE Allow Statement

```
allow user_t bin_t : file { read execute } ;
```

} *domain type*
} *object type*
} *object class*
} *permissions*

Translation: A process with type `user_t` can `read or execute files` of type `bin_t`

- All access is denied unless explicitly allowed
- see policy development course slides for more:

www.tresys.com/selinux/



Why Type Enforcement

- Configurable mandatory access control
 - flexible (not tied to a single security objective)
 - dynamic (loadable policy, on-going extension work)
 - possible to be pragmatic within a policy
 - even necessary in Linux due to legacy!
 - fine-grained access control
 - object classes and permissions, unlimited types and rules
- But...



Challenges with SE Linux

- Policies can be complex
 - legacy issues with Linux/Unix
 - large, general-purpose OS
- Flexibility comes with a price!
 - tens of object classes, over a hundred permissions
 - unlimited domain, types, object instances
 - tens of thousands of rules not uncommon
- Assurance of mechanism untested in traditional manner
 - open source helps, modestly high assurance at best (B2/EAL5)
 - certainly no worse than Linux (or other mainstream OSs)
 - in fact much better with a good TE policy

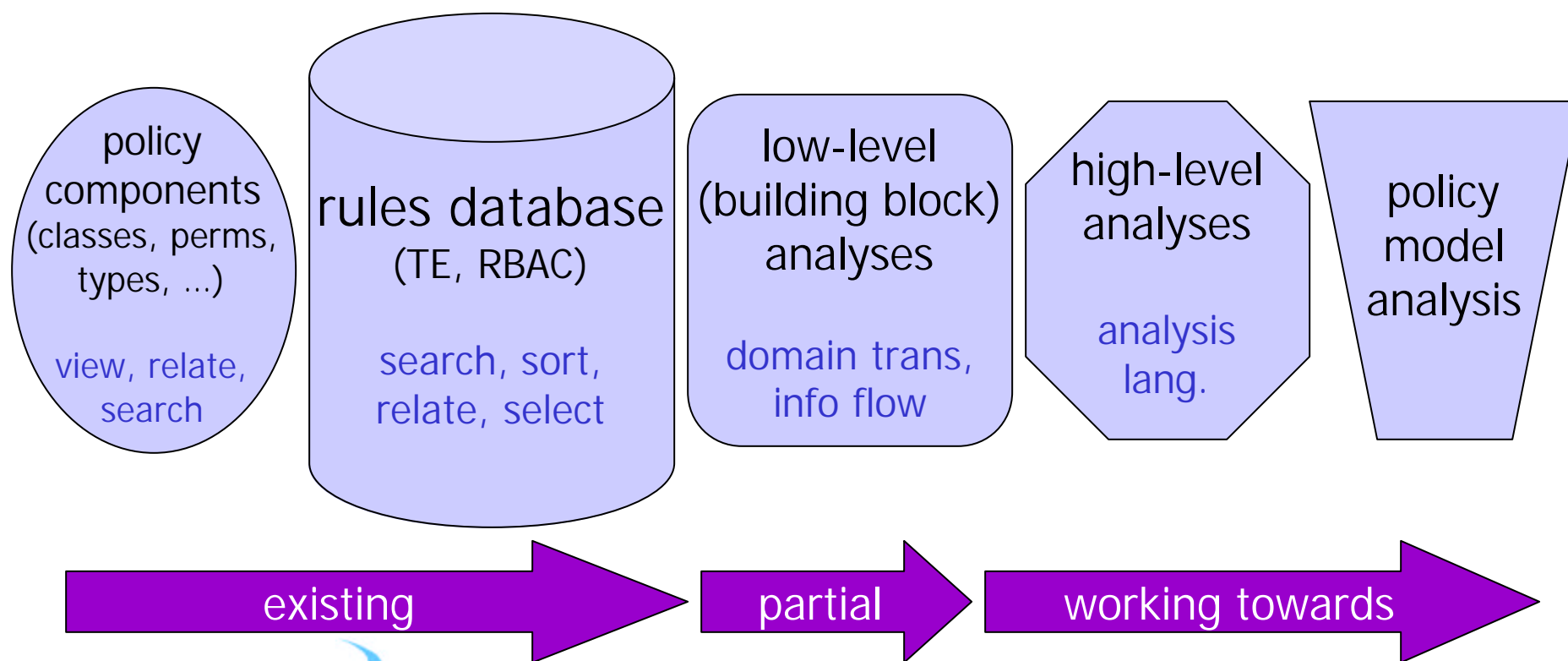


Why Analyze TE Policies?

- TE does not imply particular security properties
 - unlike most non-discretionary access control mechanisms
 - desired security properties must be demonstrated
- SE Linux is configurable and dynamic
 - given application policy must be certified
- SE Linux can have some scary policy complexity
 - large, complex OS (compared to security kernel)
 - current worse case default policy:
 - 1,000 types
 - 17,870 allow rules
 - 926 type_transition rules

APOL (Policy Analysis Tool)

- Evolutionary Development (driven by practice)
 - from basic components to abstract analyses



APOL: Policy Analysis Tool

File Search View Advanced Help

Policy Components Policy Rules Analysis policy.conf

Analysis Type

- Forward Domain Transition
- Reverse Domain Transition
- Direct Information Flow**
- Transitive Information Flow (Exper)

Analysis Options

Required parameters

Starting type: sysadm_t

Select starting type using attrib: domain

Flow direction: In Out Either Both

Optional result filters

Filter results by object class:

- blk_file
- capability
- chr_file
- dir
- fd
- fifo_file
- file

Filter end types using regular expression: ^user

Select All Clear All

New Update Info

Analysis Results

Empty Tab Results 1 Results 2 Results 3

Direct Information Flow Trees

- sysadm_t
 - user_chkpwd_t
 - user_cron_spool_t
 - user_cron_t
 - user_crontab_t
 - user_devpts_t
 - user_home_dir_t
 - user_home_ssh_t
 - user_home_t
 - user_ssh_t
 - user_su_t
 - user_t
 - user_tmp_t
 - user_tmpfs_t
 - user_tty_device_t

Direct Information Flow Data

Information flows into **sysadm_t** - from **user_chkpwd_t**

Object classes for **IN** flows:

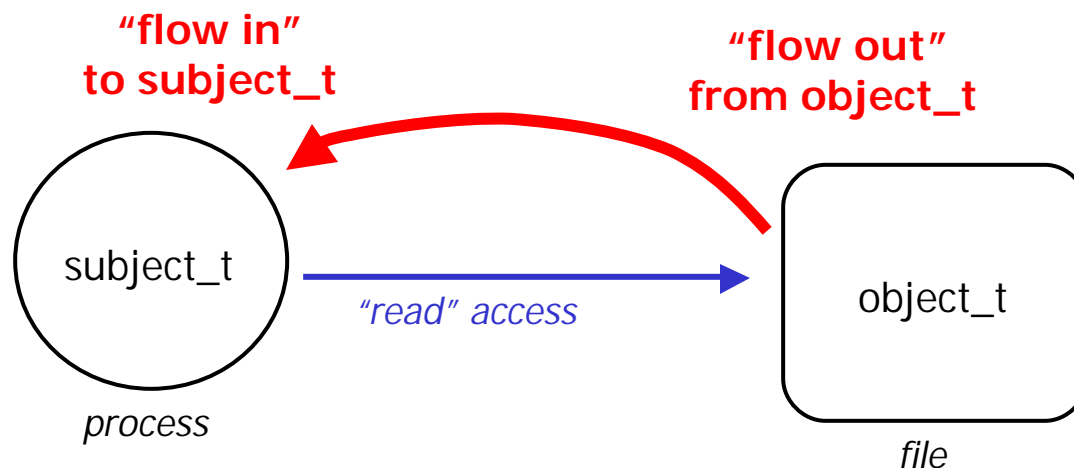
```
file
(5885) allow sysadm_t domain : { file lnk_file sock_file fifo_file } { read getattr lock ioctl };
```

Close Tab

Open a new policy Classes: 30 Perms: 122 Types: 272 Attribs: 48 TE rules: 4144 Roles: 4 RBAC rules: 7 Users: 5 v.13 -- v.15

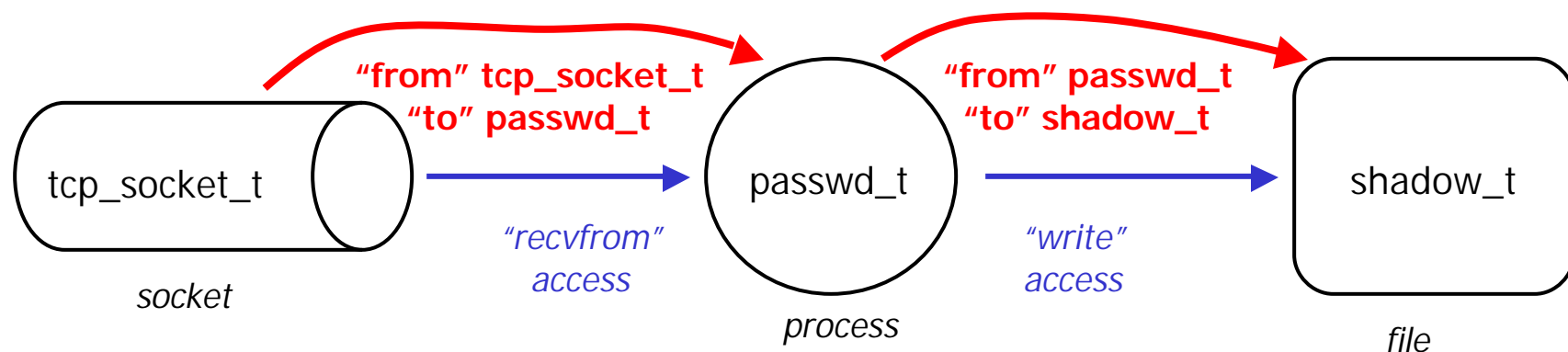
Information Flow Analysis

- low-level: direct analysis
 - map each permission to: read, write, both, none
 - from "starting type" determine: in, out, either, both
 - file class example:



Information Flow Analysis

- low-level: transitive analysis
 - determine multi-hop paths between two types
 - transitive closure and shortest path analyses
 - greatly complicated by classes and permissions
 - weighted perm maps ("overt" vs. "covert" flows)
 - filter by types ("trusted") and object class



Information Flow Analysis

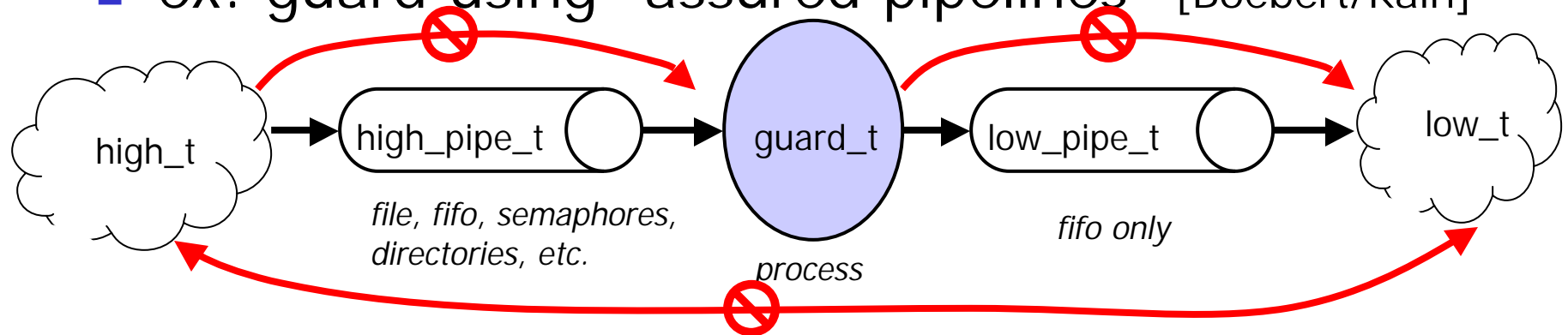
- high-level: invariant language (planned)
 - “no flow”, “only flow”, etc.
- example: high-to-low “guard” application

```
no_flow high_t low_t : * guard_t;
```

“from type” *“to type”* *“classes”* *“except these types”*

Information Flow Analysis

- policy model validation: (goal)
- ex: guard using “assured pipelines” [Boebert/Kain]



```
no_flow high_t low_t : * guard_t; //domain isolation
no_flow low_t high_t : * guard_t; //same
no_flow high_t guard_t : * high_pipe_t;
no_flow guard_t low_t : fifo low_pipe_t;
no_flow high_pipe_t low_pipe_t : * guard_t;
```



Tresys SeTools Package

- Source distribution under GPL
- Tools in package
 - **Apol**: Policy analysis/debug tool
 - **SeAudit**: audit-to-policy analyzer
 - **SePCuT**: Policy customization, basic editing
 - **SeUser**: Example user manager for SELinux
 - Misc. command line tools in progress
 - Support libraries (the “guts” of the package)



Dynamic Type Enforcement Policy

- Conditional Policy Constructions
 - Adds booleans and “ifs” to the run-time language
 - Fine-grained, TE-based control of booleans
 - For example, “audit-on” and “docked” booleans
 - Will be released this month (core SE Linux)
- Binary Policy Modules
 - support for application install without policy source
- Policy Management server
 - manage the policy as abstract, TE-protected objects
 - user mode, fine-grained protection
- All in progress, or soon to be started



QUESTIONS?

Frank Mayer
Tresys Technology
mayerf@tresys.com

www.tresys.com/selinux